

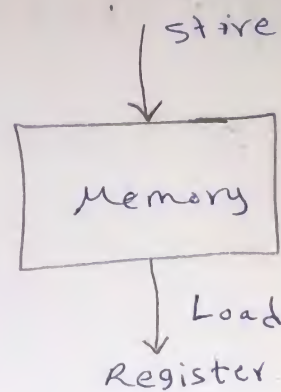
ملف 2/15

sheet 3

2/15

2 $A \times B + C \times D$

Load A
 Multiply B
 store Result
 Load C
 multiply D
 ADD Result
 store result



Result
 $A \times B$

Accumulator
 $C \times D$

$A \times B + C \times D$

3

For ($J = n-1$; $J \geq 0$; $J--$)

{

For ($K = J-1$; $K \geq 0$; $K--$)

{

if ($List[K] > List[J]$)

}

1

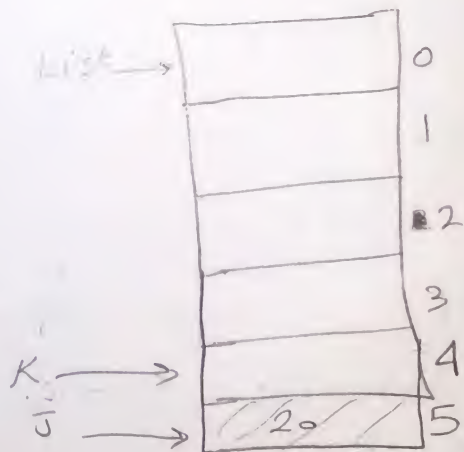
```

{
temp = List[K];
List[K] = List[J];
List[J] = temp;
}
}
}

```

Note

J ← ثابت
 K ← يتحرك لأعلى ويرى كل
 القيمة التي يشير إليها أكبر من
 K ولا لا وعلى أساس هذه القيمة
 الترتيب.



~~في كل مرة~~
 ← بعد الانتهاء من ال (loop) ننقل إلى
 جديدة فنقوم بتحريك K في مكان آخر ونستمر.
 ويحرك K كما سمع.

سرقم (۳) هو مثال لـ (insertion sort)

السؤال عايز برنامج
(assembly) يمشي الـ (Code).

Sol

Move #List, R0

Move N, R1

Subtract #1, R1

out Move R1, R2

Subtract #1, R2

Movebyte (R0, R1), R3

Inner Comparebyte R3, (R0, R2)

Branch <= 0 Next

Movebyte (R0, R2), R4

Move byte R3, (R0, R2)

Movebyte R4, (R0, R1)

R0 List

R1 N-1

R2 N-1-1

R3 20

List[K] List[J] R4 temp
(R0, R2) - R3 = 0
dst. src < 0
70

Next

Decrement R2

Branch 7=0 Inner

Decrement R1

Branch 7=0 ~~out~~ out

هناها نتيجة

المقارنة أكبر من الهدف

مخرج لأدلة ال (loop)

التي اسمها (Inner)

[4]

ORIGIN 1000

DATAWORD 300

الإشارة يتقوسا بنفس

الوظيفة لكم في أوقات مختلفة

الأدلة Assembler Directives

Move #300, 1000

Assembler

Assembler

object
code

ينفذ ال Code

معلومات إضافية ل (Assembler) عنها

[4]

5

(Subroutine) على نرى ال (Function) البرمجة.

Move # AVEC, R1

Move # BVEC, R2

Move N, R3

Call Sub

Move ~~Ro~~ R0, DOT PROD

Sub CLR R0

loop Move (R1)+, R4

Multiply (R2)+, R4

ADD R4, R0

Decrement R3

Branch 7. loop

Return

مستط (loop) قد تم

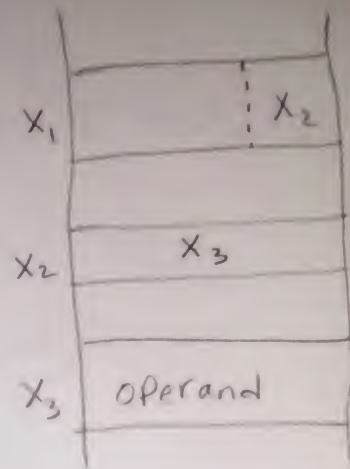
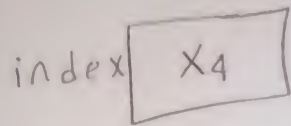
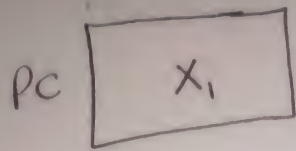
عند (CLR) لتساو

صلى ~~R0~~ الناتج الى

~~R0~~ كتابته في R0

5

6



→ في الحالات الآتية ماذا سيحدث .

Direct

$$\Rightarrow X_3 = 56$$

indirect

$$\Rightarrow X_3 = (X_2)$$

PC relative

$$\Rightarrow X_3 = X_2 + X_1 + 1$$

(PC) سينزيد بـ 1
عنايه يشار على الأمر
التالى .

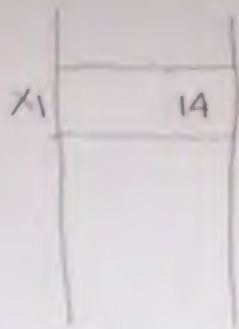
indexed

$$\Rightarrow X_3 = X_2 + X_4$$

6

7

→ ده الـ تخير في الـ اسم
عنوان
ما غير يعرف مكانه الـ (operand)



Immediate operand is 14.

Move #14, R1
نقل الـ 14 إلى الـ R1

Direct = 14

Move 14, R1
نقل الـ 14 إلى الـ R1
R1 = 14

Indirect

In memory location whose address is 14

Register

Register = 14

Register indirect

Register = 14

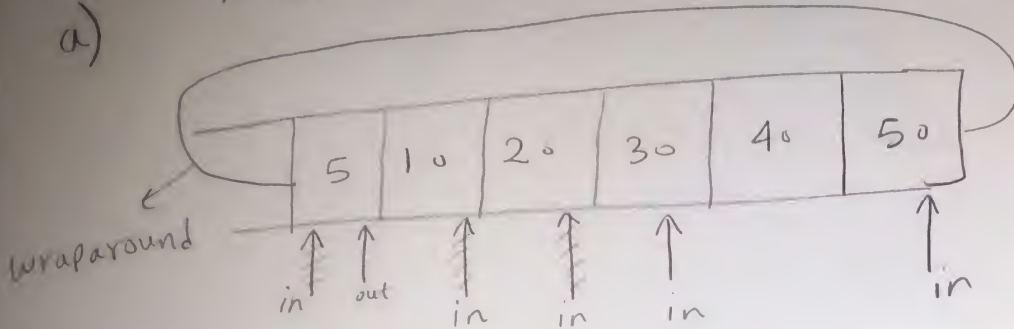
Memory loc. whose address in register 14.

7

8

FIFO

a)



← (in) مشترك جوة ال Queue -

← (out) مشترك جوة ال Queue -

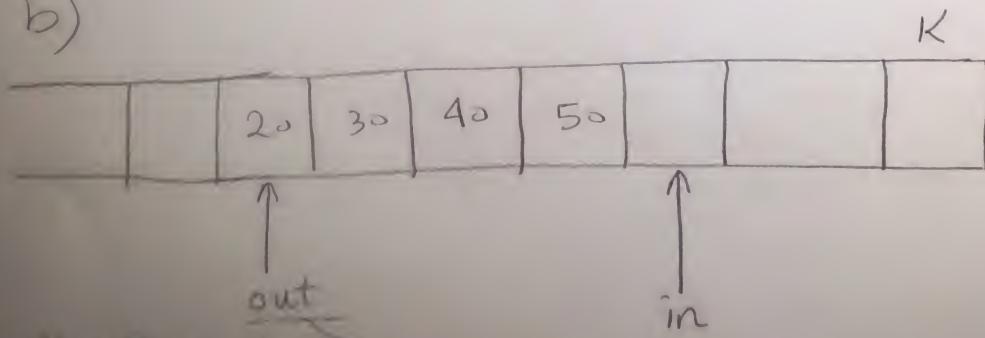
ال Queue مشترك جوة ال 10

← مشترك جوة ال Queue (50) وبقا مشترك جوة ال

مشترك جوة ال Queue (wrap around)

(a) وده حل السؤال

b)



مشترك جوة ال Queue

مشترك جوة ال

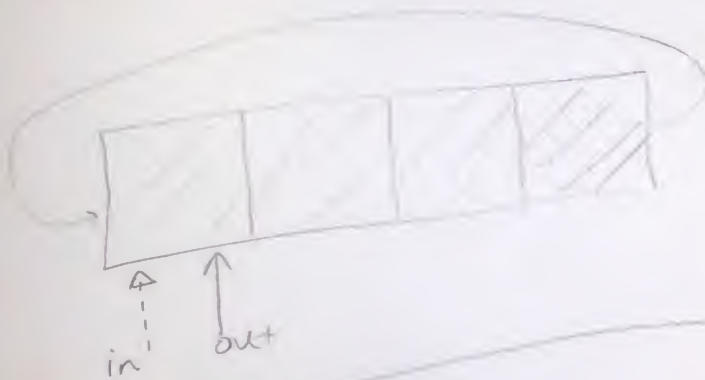
8

c)

الحل

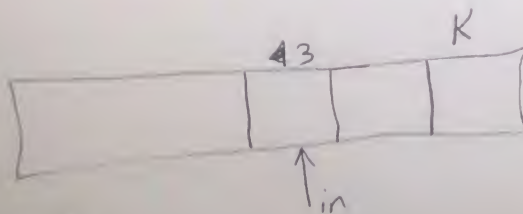
* عندما تكون $(in = out)$ يشير داء على نفس المكان
وحيث $(Queue)$ ده فاهي .

* أو (in) مستمر في ملأ الأماكن الفارغة و عمل (wrap around)
ووجد $(Queue)$ ممكن فله يقوم بشي .



$(in + 1) \mod K$

ex



if $in = 3, K = 5$

$4 \mod 5 = 4$

طالما القيمة أقل من K
الناتج يساويها .

9

d)

$Loc \leftarrow in$

$out \leftarrow (in+1) \bmod K$

← لو تباري (out) تنقول فيه ال (operation)

← مش متأكد من حلها.

9

Report

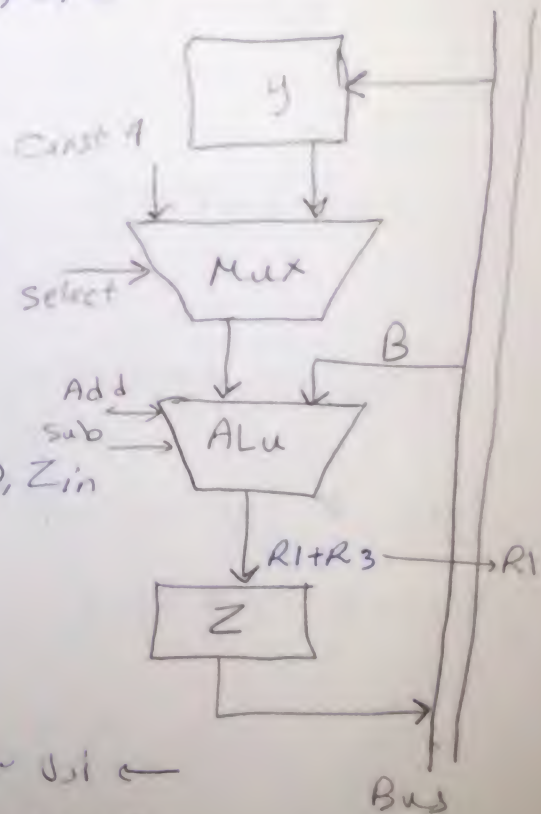
10

Sheet 4

1 WMFC

2

- 1- PC_{out}, MAR_{in}, Read, select 4, Add, Z_{in}
- 2- Z_{out}, PC_{in}, X_{in}, WMFC
- 3- MDR_{out}, IR_{in}
- 4- R₃_{out}, MAR_{in}, Read
- 5- ~~MAR~~_{out}, X_{in}, WMFC
- 6- MDR_{out}, select y, Add, Z_{in}
- 7- Z_{out}, R₁_{in}, End



(Fetch) ← اول ٣ اوضاع هما اوضاع ال
ثابتين في كل البرامج.

~~ADD~~ ADD (R3), R1

← هو (indirect) بسبب خطوة رقم (4) لأننا
خرجنا محتواه على (MAR).

← ها 7 خطوات فإذا هياخذنا
7 clock cycles
↳ execution time

3 ← WMS ← فإذا وقع عدد ال
(clock cycles)

∴ execution time = 9 clock cycle

4 delay ⇒ Bus = 0.3 ns, ALU = 2 ns
Setup → 0.2 ns, Hold time → 0

أقل وقت تأخر متبع 2 0.3 0.2

time = 2.5 ns